

## Laborator 7 PL/SQL

### Tratarea excepțiilor

- PL/SQL permite utilizatorului să capteze și să gestioneze erorile care pot apărea în timpul execuției unui program. În general, într-un bloc PL/SQL erorile ce apar sunt de 2 tipuri:
  - erori la compilare (detectate de motorul PL/SQL și comunicate programatorului, care va face corectările necesare; aceste erori nu pot fi tratate în interiorul programului)
  - erori la execuție (sunt numite excepții; trebuie specificat în program modul de tratare a acestora, caz în care se spune că excepția este tratată în program; dacă aceasta nu este tratată, atunci se va propaga în mediul din care s-a invocat programul)
- O excepție *PL/SQL* este o situație specială ce poate apărea în execuția unei bloc PL/SQL.
- O excepție poate fi gestionată:
  - în mod explicit de către utilizator (comanda RAISE);
  - în mod automat de către server, atunci când apare o eroare.
- Tratarea excepțiilor se realizează în zona EXCEPTION a unui bloc PL/SQL.

EXCEPTION

```
WHEN nume_excepție1 [OR nume_excepție2 ...] THEN
    secvența_de_instrucțiuni_1;
[WHEN nume_excepție3 [OR nume_excepție4 ...] THEN
    secvența_de_instrucțiuni_2;]
...
[WHEN OTHERS THEN
    secvența_de_instrucțiuni_n;]
END;
```

- Tipuri de excepții :
  - excepții Oracle Server predefinite (NO\_DATA\_FOUND, TOO\_MANY\_ROWS etc);
  - excepții Oracle Server nepredefinite (nu au un nume precum NO\_DATA\_FOUND, ci pot fi recunoscute doar după cod și mesaj);
  - excepții definite de utilizator.
- Informații despre erorile apărute la compilare se pot obține consultând vizualizarea USER\_ERRORS.

```
SELECT LINE, POSITION, TEXT
FROM   USER_ERRORS
WHERE  NAME = UPPER('nume');
```

*LINE* specifică numărul liniei în care apare eroarea, dar acesta nu corespunde liniei efective din fișierul text (se referă la codul sursă depus în *USER\_SOURCE*). Dacă nu sunt erori, apare mesajul *NO ROWS SELECTED*.

#### 1. Remediate rând pe rând excepțiile din următorul exemplu.

```
SET SERVEROUT ON
DECLARE
  v NUMBER;
  CURSOR c IS
    SELECT employee_id FROM employees;
BEGIN
```

```

-- no data found
SELECT employee_id
INTO v
FROM employees
WHERE 1=0;
-- too many rows
SELECT employee_id
INTO v
FROM employees;
-- invalid number
SELECT employee_id
INTO v
FROM employees;
WHERE 2='s';
-- when others
v := 's';
-- cursor already open
open c;
open c;
EXCEPTION
WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE (' no data found: ' || SQLCODE || ' - ' || SQLERRM);
WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE (' too many rows: ' || SQLCODE || ' - ' || SQLERRM);
WHEN INVALID_NUMBER THEN
    DBMS_OUTPUT.PUT_LINE (' invalid number: ' || SQLCODE || ' - ' || SQLERRM);
WHEN CURSOR_ALREADY_OPEN THEN
    DBMS_OUTPUT.PUT_LINE (' cursor already open: ' || SQLCODE || ' - ' || SQLERRM);
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE (SQLCODE || ' - ' || SQLERRM);
END;
/
SET SERVEROUT OFF

```

2. Să se creeze tabelul *error\_\*\*\** care va conține două câmpuri: *cod* de tip NUMBER și *mesaj* de tip VARCHAR2(100). Să se creeze un bloc PL/SQL care să permită gestiunea erorii „divide by zero” în două moduri: prin definirea unei excepții de către utilizator și prin captarea erorii interne a sistemului. Codul și mesajul erorii vor fi introduse în tabelul *error\_\*\*\**.

```

DROP TABLE error_***;

CREATE TABLE error_***
(cod      NUMBER,
 mesaj   VARCHAR2(100));

```

Varianta 1

```

DECLARE
    v_cod      NUMBER;
    v_mesaj    VARCHAR2(100);
    x          NUMBER;
    exceptie   EXCEPTION;

BEGIN
    x:=1;
    IF x=1 THEN RAISE exceptie;
    ELSE
        x:=x/ (x-1);
    END IF;
EXCEPTION
    WHEN exceptie THEN
        v_cod := -20001;
        v_mesaj := 'x=1 determina o impartire la 0';
        INSERT INTO error_***
        VALUES (v_cod, v_mesaj);
    END;
/
SELECT *
FROM error_***;

```

Varianta 2

```

DECLARE
    v_cod      NUMBER;
    v_mesaj    VARCHAR2(100);
    x          NUMBER;
BEGIN
    x:=1;
    x:=x/ (x-1);
EXCEPTION
    WHEN ZERO_DIVIDE THEN
        v_cod := SQLCODE;
        v_mesaj := SUBSTR(SQLERRM,1,100);
        -- mesajul erorii are dimensiune 512
        INSERT INTO error_***
        VALUES (v_cod, v_mesaj);
    END;
/
SELECT *
FROM error_***;
ROLLBACK;

```

3. Să se creeze un bloc *PL/SQL* prin care să se afișeze numele departamentului care funcționează într-o anumită locație. Dacă interogarea nu întoarce nicio linie, atunci să se trateze excepția și să se insereze în tabelul *error\_\*\*\** codul erorii -20002 cu mesajul "nu există departamente în locația data". Dacă interogarea întoarce o singură linie, atunci să se afișeze numele departamentului. Dacă interogarea întoarce mai multe linii, atunci să se introducă în tabelul *error\_\*\*\** codul erorii -20003 cu mesajul "există mai multe departamente în locația data".

Testați pentru următoarele locații: 1400, 1700, 3000.

```

SET SERVEROUTPUT ON
SET VERIFY OFF
ACCEPT p_loc PROMPT 'Dati locatia: '

DECLARE
    v_loc      dept_***.location_id%TYPE:= &p_loc;
    v_nume     dept_***.department_name%TYPE;
BEGIN
    SELECT      department_name
    INTO        v_nume
    FROM        dept_***
    WHERE       location_id = v_loc;
    DBMS_OUTPUT.PUT_LINE('In locatia '|| v_loc ||
                          ' functioneaza departamentul '||v_nume);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        INSERT INTO error_***
        VALUES (-20002, 'nu există departamente în locația data');
        DBMS_OUTPUT.PUT_LINE('a aparut o excepție ');
    WHEN TOO_MANY_ROWS THEN
        INSERT INTO error_***
        VALUES (-20003,
                'există mai multe departamente în locația data');
        DBMS_OUTPUT.PUT_LINE('a aparut o excepție ');
    WHEN OTHERS THEN
        INSERT INTO error_*** (mesaj)
        VALUES ('au aparut alte erori');
END;
/
SET VERIFY ON
SET SERVEROUTPUT OFF

```

4. Să se adauge constrângerea de cheie primară pentru câmpul *department\_id* din tabelul *dept\_\*\*\** și constrângerea de cheie externă pentru câmpul *department\_id* din tabelul *emp\_\*\*\** care referă câmpul cu același nume din tabelul *dept\_\*\*\**.

Să se creeze un bloc *PL/SQL* care tratează excepția apărută în cazul în care se șterge un departament în care lucrează angajați (**excepție internă nepredefinită**).

```

ALTER TABLE dept_***
ADD CONSTRAINT c_pr_*** PRIMARY KEY(department_id);

ALTER TABLE emp_***
ADD CONSTRAINT c_ex_*** FOREIGN KEY (department_id)
REFERENCES dept_***;

DELETE FROM dept_***
WHERE department_id=10; --apare eroarea sistem -02292

SET SERVEROUTPUT ON
SET VERIFY OFF
ACCEPT p_cod PROMPT 'Dati un cod de departament '
DECLARE
    exceptie EXCEPTION;
    PRAGMA EXCEPTION_INIT(exceptie,-02292);
    -- exceptia nu are un nume predefinit,
    -- cu PRAGMA EXCEPTION_INIT asociat erorii avand
    -- codul -02292 un nume
BEGIN
    DELETE FROM dept_***
    WHERE department_id = &p_cod;
EXCEPTION
    WHEN exceptie THEN
        DBMS_OUTPUT.PUT_LINE ('nu puteti sterge un departament in care
lucreaza salariati');
END;
/
SET VERIFY ON
SET SERVEROUTPUT OFF

```

**5.** Să se creeze un bloc *PL/SQL* prin care se afișează numărul de salariați care au venitul anual mai mare decât o valoare dată. Să se trateze cazul în care niciun salariat nu îndeplinește această condiție (**excepții externe**).

```

SET SERVEROUTPUT ON
SET VERIFY OFF
ACCEPT p_val PROMPT 'Dati valoarea: '
DECLARE
    v_val          NUMBER := &p_val;
    v_numar        NUMBER(7);
    exceptie       EXCEPTION;
BEGIN
    SELECT COUNT(*)
    INTO   v_numar
    FROM   emp_***
    WHERE   (salary+salary*NVL(commission_pct,0))*12>v_val;

```

```

IF v_numar = 0 THEN
    RAISE exceptie;
ELSE
    DBMS_OUTPUT.PUT_LINE('NR de angajati este'||v_numar);
END IF;

EXCEPTION
    WHEN exceptie THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista angajati pentru care sa se
indeplineasca aceasta conditie');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Alta eroare');
END;
/
SET VERIFY ON
SET SERVEROUTPUT OFF

```

**6.** Să se mărească cu 1000 salariul unui angajat al cărui cod este dat de la tastatură. Să se trateze cazul în care nu există angajatul al cărui cod este specificat. Tratarea excepție se va face **în secțiunea executabilă**.

```

SET VERIFY OFF
ACCEPT p_cod PROMPT 'Dati codul: '
DECLARE
    v_cod          NUMBER := &p_cod;
BEGIN
UPDATE emp_***
SET salary=salary+1000
WHERE employee_id=v_cod;
IF SQL%NOTFOUND THEN
    RAISE_APPLICATION_ERROR(-20999,'salariatul nu exista');
END IF;
END;
/
SET VERIFY ON

```

**7.** Să se afișeze numele și salariul unui angajat al cărui cod este dat de la tastatură. Să se trateze cazul în care nu există angajatul al cărui cod este specificat. Tratarea excepție se va face **în secțiunea de tratare a erorilor**.

```

SET SERVEROUTPUT ON
SET VERIFY OFF
ACCEPT p_cod PROMPT 'Dati codul: '
DECLARE
    v_cod      NUMBER := &p_cod;
    v_nume    emp_***.last_name%TYPE;
    v_sal     emp_***.salary%TYPE;

```

```

BEGIN
  SELECT last_name,salary
  INTO   v_nume,v_sal
  FROM   emp_***
  WHERE  employee_id=v_cod;
  DBMS_OUTPUT.PUT_LINE(v_nume||' '||v_sal);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR(-20999, 'salariatul nu exista');
END;
/
SET VERIFY ON
SET SERVEROUTPUT OFF

```

8. Să se creeze un bloc PL/SQL care folosește 3 comenzi SELECT. Una dintre aceste comenzi nu va întoarce nicio linie. Să se determine care dintre cele trei comenzi SELECT determină apariția excepției NO\_DATA\_FOUND.

Varianta 1 – fiecare comandă are un număr de ordine

```

SET SERVEROUTPUT ON
DECLARE
  v_localizare  NUMBER(1):=1;
  v_nume   emp_***.last_name%TYPE;
  v_sal     emp_***.salary%TYPE;
  v_job      emp_***.job_id%TYPE;

BEGIN
v_localizare:=1;
SELECT last_name
INTO   v_nume
FROM   emp_***
WHERE  employee_id=200;
DBMS_OUTPUT.PUT_LINE(v_nume);

v_localizare:=2;
SELECT salary
INTO   v_sal
FROM   emp_***
WHERE  employee_id=455;
DBMS_OUTPUT.PUT_LINE(v_sal);

v_localizare:=3;
SELECT job_id
INTO   v_job
FROM   emp_***
WHERE  employee_id=200;
DBMS_OUTPUT.PUT_LINE(v_job);

```

```

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('comanda SELECT ' || v_localizare || ' nu
retorneaza nimic');
END;
/
SET SERVEROUTPUT OFF

```

Varianta 2 - fiecare comandă este inclusă într-un subbloc

```

SET SERVEROUTPUT ON
DECLARE
  v_localizare NUMBER(1):=1;
  v_nume emp_***.last_name%TYPE;
  v_sal   emp_***.salary%TYPE;
  v_job   emp_***.job_id%TYPE;
BEGIN

```

```

  BEGIN
    SELECT last_name
    INTO   v_nume
    FROM   emp_***
    WHERE  employee_id=200;
    DBMS_OUTPUT.PUT_LINE(v_nume);
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('comanda SELECT1 nu returneaza nimic');
  END;

  BEGIN
    SELECT salary
    INTO   v_sal
    FROM   emp_***
    WHERE  employee_id=455;
    DBMS_OUTPUT.PUT_LINE('v_sal');
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('comanda SELECT2 nu returneaza nimic');
  END;

  BEGIN
    SELECT job_id
    INTO   v_job
    FROM   emp_***
    WHERE  employee_id=200;
    DBMS_OUTPUT.PUT_LINE(v_job);
  
```

```

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('comanda SELECT3 nu returneaza nimic');
END;

END;
/
SET SERVEROUTPUT OFF

```

**9.** Dați un exemplu prin care să se arate că nu este permis **saltul de la secțiunea de tratare a unei excepții, în blocul curent**.

```

DECLARE
  v_comm  NUMBER(4);
BEGIN
  SELECT ROUND(salary*NVL(commission_pct,0))
  INTO   v_comm
  FROM   emp_***
  WHERE  employee_id=455;
<<eticheta>>
  UPDATE emp_***
  SET    salary=salary+v_comm
  WHERE employee_id=200;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    v_comm:=5000;
    GOTO eticheta;
END;
/

```

**10.** Dați un exemplu prin care să se arate că nu este permis saltul la secțiunea de tratare a unei excepții.

```

SET SERVEROUTPUT ON
DECLARE
  v_comm_val  NUMBER(4);
  v_comm      emp_***.commission_pct%TYPE;
BEGIN
  SELECT NVL(commission_pct,0),
         ROUND(salary*NVL(commission_pct,0))
  INTO   v_comm, v_comm_val
  FROM   emp_***
  WHERE  employee_id=200;
  IF v_comm=0
  THEN
    GOTO eticheta;

```

```

ELSE
    UPDATE emp_***
    SET salary=salary+ v_comm_val
    WHERE employee_id=200;
END IF;
<<eticheta>>
--DBMS_OUTPUT.PUT_LINE('este ok!');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('este o exceptie!');
END;
/
SET SERVEROUTPUT OFF

```

## EXERCITII

- E1.** Să se creeze un bloc *PL/SQL* care afișează radicalul unei variabile introduse de la tastatură. Să se trateze cazul în care valoarea variabilei este negativă. Gestionaerea erorii se va realiza prin definirea unei excepții de către utilizator, respectiv prin captarea erorii interne a sistemului. Codul și mesajul erorii vor fi introduse în tabelul *error\_\*\*\**(cod, mesaj).
- E2.** Să se creeze un bloc *PL/SQL* prin care să se afișeze numele salariatului (din tabelul *emp\_\*\*\**) care câștigă un anumit salariu. Valoarea salariului se introduce de la tastatură. Se va testa programul pentru următoarele valori: 500, 3000 și 5000.  
Dacă interogarea nu întoarce nicio linie, atunci să se trateze excepția și să se afișeze mesajul “nu există salariați care să castige acest salariu”. Dacă interogarea întoarce o singură linie, atunci să se afișeze numele salariatului. Dacă interogarea întoarce mai multe linii, atunci să se afișeze mesajul “există mai mulți salariați care castiga acest salariu”.
- E3.** Să se creeze un bloc *PL/SQL* care tratează eroarea apărută în cazul în care se modifică codul unui departament în care lucrează angajați.
- E4.** Să se creeze un bloc *PL/SQL* prin care se afișează numele departamentului 10 dacă numărul său de angajați este într-un interval dat de la tastatură. Să se trateze cazul în care departamentul nu îndeplinește această condiție.
- E5.** Să se modifice numele unui departament al cărui cod este dat de la tastatură. Să se trateze cazul în care nu există acel departament. Tratarea excepție se va face în secțiunea executabilă.
- E6.** Să se creeze un bloc *PL/SQL* care afișează numele departamentului ce se află într-o anumită locație și numele departamentului ce are un anumit cod (se vor folosi două comenzi *SELECT*). Să se trateze excepția *NO\_DATA\_FOUND* și să se afișeze care dintre comenzi a determinat eroarea. Să se rezolve problema în două moduri.
- E7.** Adaptați cerința exercițiului 5 pentru diagrama proiectului prezentată la materia Baze de Date din anul I. Rezolvați acest exercițiu în *PL/SQL*, folosind baza de date proprie.